Bustan Raluca - Cristina

334AA

Tema 2

1. Protocolul de intrare:

   Stari : `**fetch_header** – salveaza numarul de operanzi si codul operatiei + verifica daca nr de operanzi e zero, caz in care trimite in eroare.

   `**fetch_payload** – am folosit i2 pentru indexarea la citire si j pentru incrementarea pe frontul de ceas crescator. Cand s-au parcurs toti operanzii (i2+1=nr. operanzi) se trece in starea urmatoare.

   `**check_mod** – alege starea urmatoare in functie de modul operandului . Mod=01 trece la cautarea in memorie, altfel incrementeaza indexul si verifica din nou. Cand indexul devine egal cu nr de operanzi se trece la calculul operatiei.

2. Protocolul AMM:
   Stari: `**decode** – in primul if pun read si address pe 1 cat timp waitrequest nu e 0. Cand devine 0 si response-ul este 00 fac citirea efectiva din memorie si trec modul pe 00, intrucat am obtinut o valoare. Pt alte valori ale response-ului rezulta o eroare.

3. Calculul operatiei:
   Stari: `**execute**: Precizari:

   La AND am tratat doua cazuri astfel: result este initial 0. Cand incepe operatia AND se pune in rezultat prima valoare a payloadului iar apoi se face & cu restul operanzilor.

   La NOT am trunchiat rezultatul la 8 biti.

   La cazurile cu operatii cu un singur operand am verficat daca primesc un singur operand, altfel eroare. Similar pentru cazurile cu doi operanzi.

   La NEG am facut SBB1 0, payload si am trunchiat rezultatul la 8 biti.

4. Protocolul de iesire:
   Stari: ` **store_header**: aici concatenez bitul de eroare cu codul operatiei pt obtinerea headerului.

   `**store_payload**: pun rezultatul operatiei respective si consider erorile.

4. Cazurile de eroare: Pentru erori am considerat o stare (error) in care setez variabila err 1 cand o alta stare trimite in eroare.